ANALISA PERFORMANSI TEKNIK *CHANNEL CODING* DAN *DECODING* MENGGUNAKAN *CONVOLUTIONAL CODE* DAN ALGORITMA VITERBI

TUGAS AKHIR

Oleh :

GALIH JOKO HERIANTO
10355023153

JURUSAN TEKNIK ELEKTRO
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM
RIAU
PEKANBARU
2011

ANALISA PERFORMANSI TEKNIK *CHANNEL CODING* DAN *DECODING* MENGGUNAKAN *CONVOLUTIONAL CODE* DAN ALGORITMA VITERBI

GALIH JOKO HERIANTO
10355023153

Tanggal Sidang : 04 Februari 2011

Perioda Wisuda : Juli 2011

Jurusan Teknik Elektro

Fakultas Sains dan Teknologi

Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Channel coding berfungsi untuk menjaga informasi atau data digital dari *error* yang mungkin terjadi selama proses transmisi dengan cara menambahkan bit *redundansi* (tambahan) ke dalam data yang akan dikirimkan. Pada penelitian ini, dilakukan simulasi dan analisa terhadap performansi teknik *channel coding* dan *decoding* menggunakan *convolutional code* dan algoritma viterbi pada kanal dengan AWGN *(Additive White Gaussian Noise)* menggunakan bahasa pemograman Matlab 7.1 serta menggunakan modulasi QPSK. Dalam simulasi diperoleh hasil perbandingan performansi antara sistem tanpa menggunakan *channel coding* dan *decoding* dengan sistem yang menggunakan teknik *channel coding* (*convolutional code*) dan *decoding* (algoritma viterbi). Hasil simulasi ditampilkan dalam bentuk grafik SNR (*Signal to Noise Ratio*) terhadap BER (*Bit Error Rate*).

Standar BER yang digunakan adalah standar minimum untuk transmisi suara yaitu 10^{-3}. Pada komunikasi tanpa *covolutional code* dan algoritma viterbi, sistem tidak dapat mencapai BER dengan standar minimum yang telah ditetapkan. Dengan menerapkan *covolutional code* dan algoritma viterbi hanya dibutuhkan SNR 5dB untuk mendapatkan kualitas komunikasi yang memenuhi standar. Dengan penambahan *convolutional code* dan algoritma viterbi dapat menghemat daya karena SNR yang dibutuhkan lebih kecil.

Kata kunci : Algoritma Viterbi, AWGN, BER, *Channel Coding*, *Channel Decoding*, *Convolutional Code*, Performansi, QPSK, SNR

ANALISA PERFORMANSI TEKNIK *CHANNEL CODING* DAN *DECODING* MENGGUNAKAN *CONVOLUTIONAL CODE* DAN ALGORITMA VITERBI

GALIH JOKO HERIANTO
10355023153

Date of Final Exam : 04 February 2011

Graduation Ceremony Priod : July 2011

Electrical Engineering Department

Faculty of Sciences and Technology

State Islamic University of Sultan Syarif Kasim Riau

ABSTRACT

Channel coding serves to keep the digital information from errors that might occur during the transmission process by adding a redundancy bit into the data that will be be transmitted. In this research, simulation and performance analysis of channel coding and decoding techniques use the convolutional code and Viterbi algorithm in the channel with AWGN (Additive White Gaussian Noise) by Matlab 7.1. In addition, the model system was modulated by QPSK modulation before being transmitted to the channel. The simulation result describe the comparison of the systems without and with channel coding (convolutional code) and decoding (viterbi algorithm). The result were shown by the SNR (Signal to Noise Ratio) to BER (Bit Error Rate) graphs.

BER standard used was the minimum standard for voice transmission that is 10^{-3}*. In the communication without covolutional code and Viterbi algorithm, the system could not achieve the BER with the requirement of minimum standard. By applying the convolutional code and viterbi algorithm, 5dB SNR just needed to get the quality of communication that meets the standard. Moreover, it could reduce the power consumption due to the smaller SNR.*

Keywords : Algoritma Viterbi, AWGN, BER, Channel Coding, Channel Decoding, Convolutional Code, Performansi, QPSK, SNR

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Komunikasi pada dasarnya proses untuk menyampaikan pesan dari suatu tempat ke tempat yang lain, Komunikasi yang dikehendaki berupa data atau bit informasi yang dikirim sama dengan data atau bit informasi yang diterima. Suatu hal yang harus diperhatikan dalam proses pengiriman bit informasi adalah *error* yang sering timbul pada saat pengiriman bit informasi. Kemungkinan terjadinya *error* pada saat pengiriman cukup besar, sehingga dapat mengakibatkan bit informasi yang dikirimkan tidak sama dengan bit informasi yang diterima. Teori ini ditemukan oleh Claude E. Shannon (1916-2001) dia dikenal sebagai bapak dari teori informasi dengan konsep informasi dan *information entropy*.

Cara memperbaiki *error* tersebut yaitu dengan menggunakan *Convolutional Code* yang merupakan salah satu code yang bisa mendeteksi dan mengkoreksi *error* secara otomatis. Pengoreksi *error* ini lebih ditujukan pada sistem komunikasi digital, dimana informasi yang diolah dalam bentuk bit ini dikirimkan melalui suatu kanal.

Suatu *convolutional coding* ini akan menghasilkan *output* yang nilainya bergantung pada *input* bit informasi sebelumnya, sehingga dalam implementasinya memerlukan memori. *Decoding* memproses urutan kata sandi yang diterima sehingga dihasilkan urutan perkiraan informasi yang diharapkan sama dengan urutan informasi asli. Salah satu teknik *decoding* pada *convolutional*

adalah *Maximum-Likelihood Decoding* dengan Algoritma Viterbi yang diperkenalkan oleh Viterbi tahun 1967 menggunakan struktur *code trellis* dan menentukan perkiraan kemiripan maksimum dari urutan yang ditransmisikan yang mempunyai matrik terkecil.

Untuk mensimulasikan peformansi *convolutional code* dan Algoritma Viterbi ini menggunakan Matlab 7.1. Namun demikian tugas akhir ini hanya menganalisa performansi teknik *channel coding* dan *decoding* menggunakan *convolutional code* dan Algoritma Viterbi serta modulasi yang digunakan yaitu QPSK (*Quadrature Phasa Shift Keying*), karena pada penelitian sebelumnya telah menggunakan modulasi lainnya yaitu BPSK, QPSK, dan 16PSK, dengan demikian saya hanya mememilih salah satu dari modulasi tersebut sebagai contohnya, guna untuk tidak memperluas dari batasan masalahnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas maka penulis merumuskan permasalahannya yaitu mensimulasikan performansi teknik *channel coding* dan *decoding* menggunakan *convolutional code* dan Algoritma Viterbi serta performansi tanpa menggunakan *convolutional code* dan Algoritma Viterbi.

1.3 Batasan Masalah

Dalam tugas akhir ini pembahasan dibatasi pada hal-hal sebagai berikut :

1. Teknik *channel coding* dengan menggunakan *convolutional code* tanpa menggunakan *block code*.

2. *Convolutional code* menggunakan *trellis diagram*.
3. Teknik *channel decoding* dengan menggunakan Algoritma Viterbi (*Maximum-Likelihood Decoding*).
4. Modulasi yang digunakan hanya QPSK.
5. Untuk menganalisanya menggunakan bahasa pemograman matlab versi 7.1.
6. Hasil analisa yang ditampilkan berupa grafik BER (*Bit Error Rate*) dan SNR (*Signal to Noise Ratio*).

1.4 Tujuan Penulisan

Adapun tujuan pada tugas akhir ini adalah untuk mengetahui performansi teknik *channel coding* dan *decoding* menggunakan *convolutional code* dan Algoritma Viterbi.

1.5 Metodologi Penelitian

1. Studi Literatur

   Sebagai bahan acuan untuk mengkaji teori-teori dasar dan teori pendukung yang berupa buku-buku, jurnal-jurnal, *internet resource* atau majalah-majalah yang menunjang masalah *channel coding* dan *decoding*, *convolutional code* dan Algoritma Viterbi.

2. Pemodelan dan Simulasi

   Pemodelan transmisi teknik *channel coding* dan *decoding* menggunakan *convolutional code* dan Algoritma Viterbi, serta melakukan simulasi dari

permodelan tersebut. Simulasi yang digunakan berupa bahasa pemograman komputer untuk mengitung dan menganalisa data.

3. Analisa

   Yang akan dianalisa dari teknik *channel coding* dan *decoding* menggunakan *convolutional code* dan algoritma Viterbi, berdasarkan pada hasil simulasi.

1.6 Sistematika Penulisan

Adapun sistematika penulisan tugas akhir ini sebagai berikut :

BAB I PENDAHULUAN

Berisi tentang deskripsi umum isi tugas akhir yang meliputi : latar belakang masalah, rumusan masalah, batasan masalah, tujuan penulisan, metodologi penulisan dan sistematika penulisan.

BAB II LANDASAN TEORI

Membahas tentang pengertian teknik *channel coding* dan *decoding*, *convolutional code*, Algoritma Viterbi, AWGN dan juga tentang Matlab sebagai program yang akan digunakan untuk simulasi.

BAB III PERANCANGAN DAN PEMODELAN

Berisi perancangan program simulasi pada Matlab

BAB IV HASIL DAN ANALISA

Berisi hasil simulasi dan analisa berdasarkan hasil simulasi tersebut.

BAB V KESIMPULAN DAN SARAN

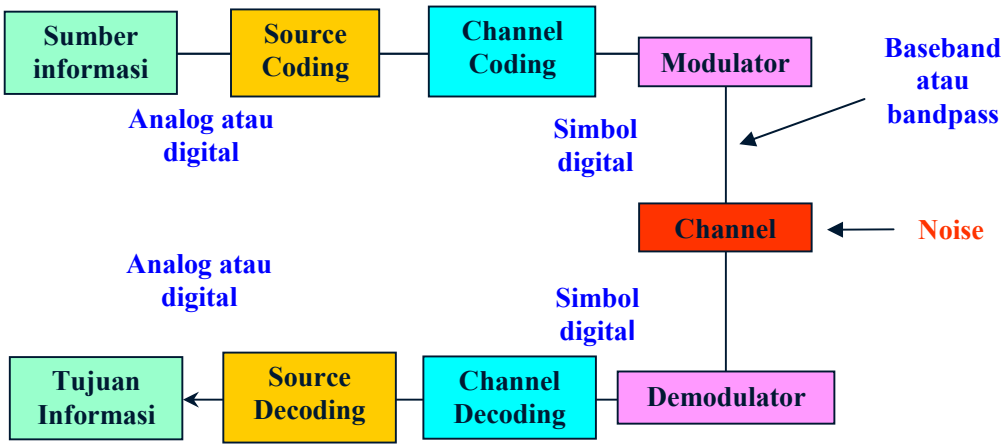
Berisi kesimpulan dan saran.

BAB II

LANDASAN TEORI

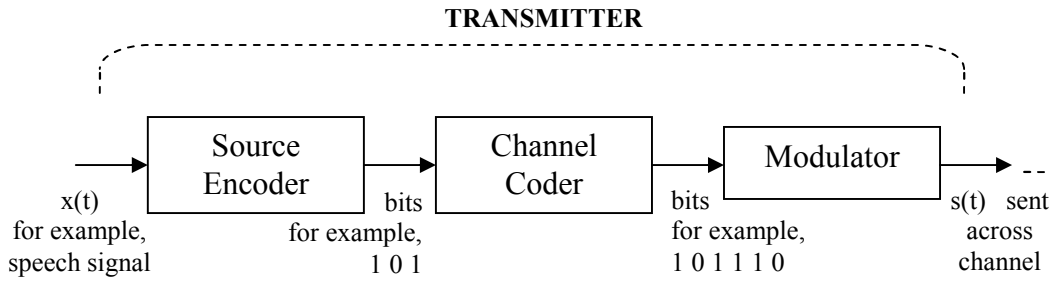
2.1 *Channel Coding Dan Decoding*

Untuk mengirimkan suatu informasi dari pengirim ke penerima, dilakukan beberapa proses dahulu terhadap informasi tersebut. Salah satunya adalah proses *channel coding*. Tahapan proses *channel coding* dalam sistem transmisi dapat dilihat dalam diagram blok sistem transmisi berikut :



Gambar 2.1 Diagram Blok Sistem Transmisi

Channel coding berfungsi untuk menjaga informasi atau data digital dari *error* yang mungkin terjadi selama proses transmisi dengan cara menambahkan bit redundansi (tambahan) ke dalam data yang akan dikirimkan. *Channel code* yang digunakan untuk mendeteksi *error* disebut *error detection code*, sedangkan yang mampu untuk mengoreksi *error* disebut *error correction code.*



Gambar 2.2 *Channel Coding*

Tujuan utama dari teknik *error* deteksi dan koreksi ini adalah untuk memperbaiki performansi sistem transmisi data digital. Dengan menambahkan bit redundansi ke dalam data yang akan dikirim maka akan meningkatkan rate transmisi atau dengan kata lain menambah *bandwidth* yang dibutuhkan jika data rate dari data aslinya diinginkan tetap.

Channel coding beroperasi pada data digital dengan mengkodekan sumber informasi ke dalam urutan kode untuk ditransmisikan melalui kanal. Ada dua macam tipe dasar *channel coding* yaitu *block code* dan *convolutional code*.

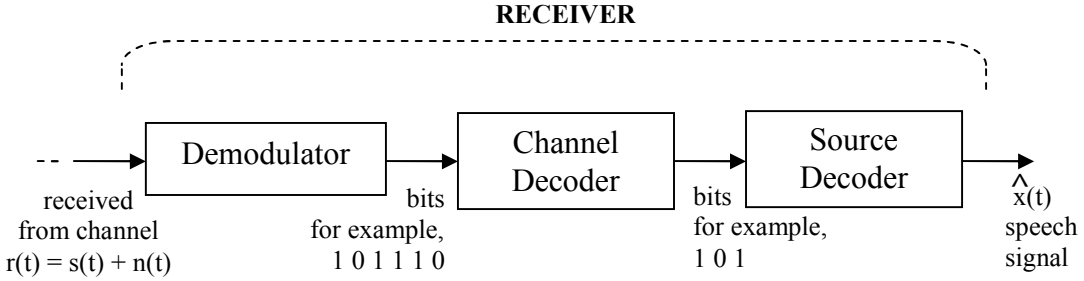
1. *Block Code*

   Block Code merupakan salah satu kode yang bersifat FEC *(Forward Error Correction*) yang mampu untuk mendeteksi dan mengkoreksi error tanpa meminta proses transmisi ulang.

2. *Convolutional Code*

   Convolutional Code merupakan jenis kode yang memiliki perbedaan mendasar dari *block code* dimana urutan bit informasi tidak dikelompok-kelompokkan dalam blok-blok yang berbeda sebelum dikodekan.

Channel Decoding berfungsi untuk memperkirakan *input* dari *coding* (informasi yang dikirimkan) dengan menggunakan aturan atau metoda tertentu yang menghasilkan kemungkinan jumlah *error* paling minimum. Dengan mengingat bahwa urutan bit kode memiliki hubungan khusus satu-satu dengan urutan bit informasinya. Urutan bit informasi dan kodenya secara unik dapat direpresentasikan pada *trellis diagram*.



Gambar 2.3 *Channel Decoding*

2.2 *Convolutional Code*

Convolutional code merupakan salah satu jenis kode yang bisa mendeteksi dan mengoreksi *error* secara otomatis dan memiliki perbedaan mendasar dari *block code* dimana urutan bit informasi tidak dikelompok-kelompokkan dalam blok-blok yang berbeda sebelum dikodekan.

Convolutional Code berbeda dari *Block Code* yaitu *Convolutional Code* dengan *n output* pengkode pada suatu waktu tertentu tidak hanya tergantung pada *k input* pada waktu itu tetapi tergantung juga *m* blok masukan sebelumnya. *Convolutional Code* (*n, k, m*) bisa diimplementasikan dengan suatu rangkaian linear *k* (*input*), *n* (*output*) dan *m* (*memory*). *n* dan *k* adalah bilangan bulat kecil

dengan rate kode yang didefinisikan $R = k/n$ atau $k < n$, tetapi *memory* harus dibuat cukup besar untuk mencapai kemungkinan *error* terkecil.

Dalam masalah khusus ketika $k = 1$, urutan pesan tidak dibagi menjadi blok-blok dan bisa diproses dengan terus-menerus. *Convolutional code* dengan batas panjang (*constraint length*) n_A terdiri dari m tingkat *shift register*, n penjumlah modulus dua dan multiplekser untuk menserialkan keluaran pengkode menjadi urutan kode tunggal. *Input* data ke pengkode disebut urutan pesan, digeser ke dalam dan sepanjang *shift register* k bit satu kali waktu, dan *output* pengkode diperoleh dengan melakukan *convolutional* dari urutan pesan dan urutan pembangkit kode. Jumlah bit *output* untuk tiap k bit *input* adalah n bit, dengan kode ratenya $R_c = k/n$. Parameter N disebut panjang *constraint* dan menunjukkan jumlah bit data *input*.

Convolutional Code ini dibentuk dengan menambah informasi tambahan (*parity*) berdasarkan bit *input*an ($u = u_1, u_2, ...u_i$) yang sedang diproses dan di*convolutional*kan dengan generator $g^{(i)}$ untuk m kode data sebelumnya, dimana m adalah panjang *memory* atau *shift register* dari kode. Jika panjang *memory* m, maka jumlah *state memory* (isi memori) adalah 2^m.

Coding dalam *convolutional* ini merupakan suatu metode penyandian yang diterapkan pada data, dimana data tersebut mempunyai peluang untuk terganggu oleh *Gaussian noise*. *Gaussian noise* adalah suatu *noise* yang mempunyai distribusi normal. *Convolutional coding* menggunakan suatu *shift register* yang berhubungan dengan logika kombinatorial dengan menggunakan penjumlahan modulus dua.

Tabel 2.1 Tabel kebenaran penjumlahan modulus dua

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Beberapa istilah dalam *convolutional code* adalah sebagai berikut :

1. *Generator Matriks*

   Generator matriks untuk *convolutional code* adalah *semi-finite* karena panjang *input* adalah *semi-finite*.

2. *Generator polynomial*

   Berfungsi untuk membangkitkan *polynomial* sesuai dengan urutan bit data *input*.

3. *Logic Table*

   Tabel kebenaran berfungsi untuk menunjukan *output* dari *coding* jika diberi urutan bit *input*.

4. *State Diagram*

   State diagram digunakan untuk merepresentasikan proses pengkodean yang berbentuk diagram sederhana. Diagram ini akan menunjukkan kemungkinan keadaan dan transisinya dari satu keadaan ke keadaan lainnya.

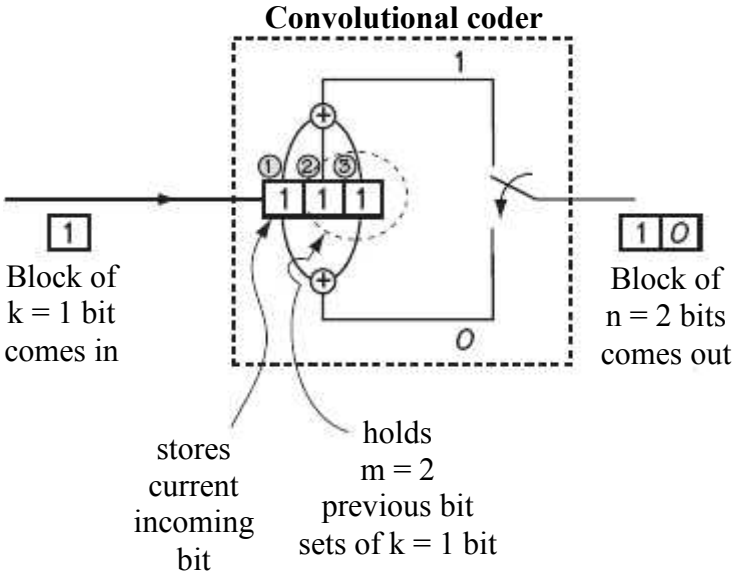
5. *Tree Diagram*

   Menunjukan struktur *coding* dalam bentuk diagram pohon dengan cabang-cabangnya menunjukan variasi keadaan dan *output* yang mungkin terjadi.

6. *Trellis Diagram*

   Merupakan bentuk penyederhaan dari diagram pohon yang merupakan representasi dari *output encoder* jika diberi *input* secara lebih kompak.

Pada *convolutional code* ini, bit data diberikan pada laju $k = 1$ bit per satuan waktu. Simbol *output* pada laju $m = 2k = 2$ simbol per satuan waktu. Bit *input*-nya akan stabil selama siklus pengkodean. Sebuah gambar akan membuatnya lebih jelas, jadi perhatikanlah Gambar 2.4 di bawah ini :



Gambar 2.4 Contoh pada *Convolutional Code* [carl, 2001].

Panah pada tampilan sebelah kiri dimana bit *input* dengan $k = 1$, bit *input* dalam 1 bit per satuan waktu. Kotak dengan garis petunjuk $k = 1$ bit berjalan ke dalam yang artinya untuk menampilkan 3 bit *shift register*. Rangkaian *convolutional code* di atas menggunakan pola C (2, 1, 2) dengan kode rate R = ½.

Time	Input bit	Shift register Contents			Output bits bit 1	bit 2
		①	②	③		
0	-	0	0	0		
1	1	1	0	0	1	1
2	0	0	1	0	1	0
3	1	1	0	1	0	0
4	0	0	1	0	1	0
5	0	0	0	1	1	1

Gambar 2.5 Contoh Sistem Kerja *Convolutional Code* [carl, 2001]

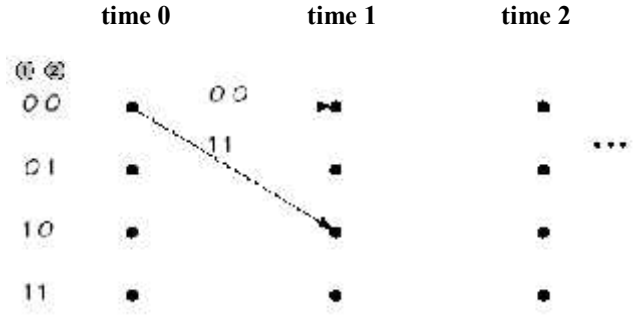
Pada Gambar 2.5 sebelum dimulai pada *time* 0 di *shift register*-nya diberi bit 0 seluruhnya, untuk *time* 1 dimana *input* bit 1 diletakkan di posisi pertama pada *shift register*, selanjutnya 0 0 pada posisi kedua dan ketiga, jadi *shift register*-nya yaitu 1 0 0. Untuk hasil bit *output* pertama adalah 1 + 0 + 0 (Modulus 2) = 1 dan bit *output* kedua adalah 1 + 0 (modulus 2) = 1, jadi hasil bit *output*-nya 1 1, begitu seterusnya pada *time* berikutnya.

Decoding dalam *convolutional* berfungsi untuk memperkirakan *input* dari *coding* (informasi yang dikirimkan) dengan menggunakan aturan atau metoda tertentu yang menghasilkan kemungkinan jumlah *error* (kesalahan) paling minimum. Dengan mengingat bahwa urutan bit kode memiliki hubungan khusus satu-satu dengan urutan bit informasinya. Urutan bit informasi dan kodenya secara unik dapat direpresentasikan pada *trellis diagram*. Suatu data di*convolutional*kan kemudian ditransmisikan ke dalam *noisy channel*. Operasi *convolutional* ini

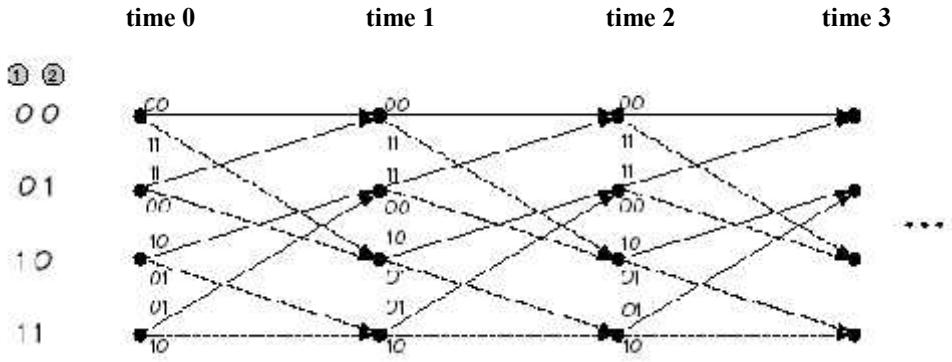
mengkodekan beberapa informasi *redundant* (tambahan) ke dalam sinyal transmisi. Teknik *convolutional decoding* ini ada banyak macamnya, salah satu metoda yang paling populer adalah Algoritma Viterbi. Algoritma ini pertama kali diperkenalkan oleh A.J. Viterbi. Untuk men*decoding*kan data yang dikodekan secara *convolutional* tersebut digunakan Algoritma Viterbi.

2.3 Algoritma Viterbi

Algoritma Viterbi ini berfungsi sebagai pengkoreksi dan pendeteksi kesalahan yang terjadi pada data setelah dilakukan penyandian dengan *convolutional coding*. Prinsip dasar dari algoritma Viterbi ini adalah metode maksimum *likelihood* dengan pengetahuan akan *trellis diagram*.



Gambar 2.6 *Trellis Diagram Partial* [carl, 2001]



Gambar 2.7 *Trellis Diagram Complete* [carl, 2001]

Pada dasarnya algoritma ini membandingkan bit yang diterima pada waktu $t = t_1$ dengan seluruh *path* pada waktu yang sama. Pada waktu $t = t_1$ tersebut akan dibandingkan nilai korelasi maksimumnya (*best matric*) atau nilai minimum *distance* dan *path* yang dipilih disebut *the surviving path*.

Jika dinyatakan dalam keadaan S_j pada waktu i pada node diagram trellis sebagai $S_{j,i}$ maka setiap node pada *trellis diagram* dapat ditunjukkan oleh nilai $V(S_{j,i})$. Nilai-nilai node pada *trellis diagram* dapat dihitung dengan cara berikut :

1. Tentukan nilai $V(S_{0,0}) = 0$ dan $i = 1$, menghitung *partial path metric* untuk jalur tunggal memasuki setiap keadaan. Menyimpan jalur (*survivor*) dan matriknya untuk setiap keadaan.
2. Pada waktu i, menghitung *partial path matric* untuk semua jalur yang memasuki keadaan dengan menambahkan *branch matric* yang memasuki keadaan tersebut dengan matrik yang terhubung *survivor* pada waktu sebelumnya. Untuk setiap keadaan, menyimpan jalur dengan matrik terkecil (*survivor*) termasuk matriknya dan menghilangkan semua jalur lain.
3. Tentukan $V(S_{j,i})$ sebesar nilai bagian jalur terkecil yang masuk ke node yang berkorespondensi dengan keadaan S_j pada waktu i.
4. Jika $l < L + m$, dimana L adalah jumlah *input* segmen kode (k bit untuk tiap segmen) dan m adalah panjang dari *shift register coding* yang terpanjang, maka nilai berubah menjadi $i = i + 1$ dan kembali ke langkah 2. [Esha Ganesha, UGM, 2007].

Algoritma Viterbi digunakan untuk memperkokoh pengiriman sinyal digital lewat kanal komunikasi nirkabel yang kurang baik. Hal ini dilakukan dengan penambahan bit *redundant* (tambahan) pada data yang merupakan kombinasi dari data sebelumnya, sehingga *decoding* dapat memprediksikan dan memperbaiki bit yang berubah.

Proses kerja Viterbi Pada penerima, *Decoding* Viterbi berusaha mengembalikan sinyal yang salah pada saat transmisi ke sinyal yang benar dengan menyimpan beberapa data sebelumnya, mengkalkulasi 'jarak konstelasi' antar data yang berurutan, dan memperkirakan data yang paling mungkin diterima sehingga bit yang salah dapat dideteksi dan diperbaiki.

2.4 Performansi *Convolutional Code*

Pada dasarnya kita harus mengetahui bagaimana teknik *channel coding* dan *decoding* dalam *convolutional code* ini bekerja. Teknik *channel coding* dan *decoding* dalam *convolutional code* ini bekerja sebagai pendeteksi dan pengoreksi bit *error*, jadi kita harus bisa menentukan seberapa baik *convolutional code* dalam mendeteksi dan mengoreksi bit yang *error*. Standar BER yang digunakan adalah standar minimum untuk transmisi suara yaitu sepuluh pangkat minus tiga (10^{-3}). [Teddy, 2007].

Besarnya kemampuan dalam mengoreksi bit *error* dari suatu kode, itu tergantung dari kode tersebut. Secara umum *convolutional code* bisa mengoreksi *e* bit *error*, dimana $k = 2^k$ dan nilai *e* dapat dirumuskan sebagai berikut :

$$e = \left[\frac{d_{\min} - 1}{2} \right] \qquad \textbf{(2.1)}$$

dimana : d_{min} = jarak minimum dari kode.

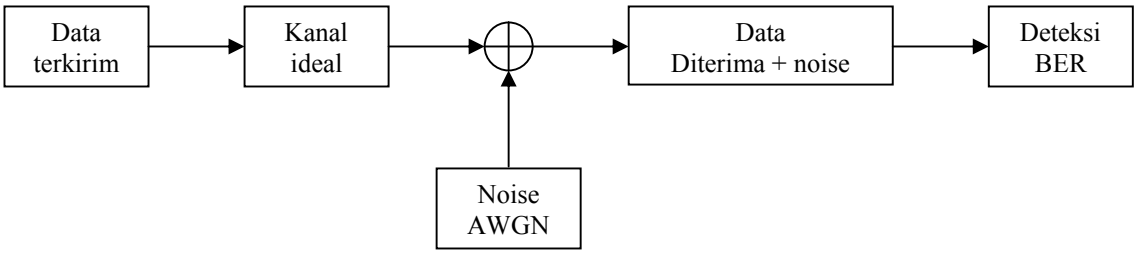
Karena *convolutional code* ini menggunakan pola C (2, 1, 2), maka d_{min} untuk convolutional code ini adalah 5 total jarak dari *all zeroes* (2 + 1 + 2) pada *trellis diagram*.

Kunci untuk menentukan seberapa baik *trellis decoding* untuk mengoreksi bit *error* adalah d_{min}. d_{min} merupakan jarak yang paling terkecil dalam menentukan suatu *trellis*.

2.5 AWGN

Kanal AWGN (*Additive White Gaussian Noise*) adalah kanal ideal yang hanya memiliki *noise* AWGN di dalamnya dan tipe kanal komunikasi digital yang paling mudah dianalisa. Disebut kanal ideal karena kanal ini tidak menyebabkan *distorsi* (perubahan bentuk sinyal) pada sinyal yang dikirim, artinya kanal ideal memiliki *bandwidth* tidak terbatas dan respon frekuensinya tetap untuk semua frekuensi.

Noise AWGN adalah noise yang pasti terjadi dalam setiap jaringan *wireless*, memiliki sifat *additive* artinya *noise* ini dijumlahkan dengan sinyal, sifat *white* artinya *noise* tidak tergantung dari frekuensi operasi sistem dan memiliki rapat daya yang konstan, dan sifat *gaussian* artinya besarnya tegangan noise memiliki rapat peluang terdistribusi *gaussian*.



Gambar 2.8 Kanal AWGN

Dalam kanal ini diasumsikan tidak ada *distorsi* atau pengaruh lainnya selain penambahan *noise white Gaussian* seperti yang ditunjukkan pada Gambar 2.8 .

2.6 MATLAB

Matlab (*matrix laboratory*) adalah sebuah *software programming* yang bekerja dengan konsep matrik dan memiliki pustaka fungsi matematika dan rekayasa yang super lengkap serta fungsi dan visualisasi baik 2D maupun 3D.

Matlab merupakan sebuah *software* yang secara sederhana dapat dianalogikan sebagai sebuah kalkulator yang kompleks. Matlab dapat melakukan operasi sederhana matematika seperti penambahan, pengurangan, perkalian, dan pembagian. Matlab juga dapat menangani bilangan komplek, akar dan pangkat, logaritma, trigonometri, serta dapat diprogram, artinya dapat mengerjakan fungsi tertentu, meng-*input*-kan dan meng-*output*-kan data, menjalankan dan menyimpan sederetan perintah untuk menjalankan dan menyimpan sederetan perintah untuk melakukan perhitungan secara otomatis. Salah satu kelebihan yang paling

menonjol dari Matlab adalah Matlab menyediakan *tool* penyelesaian masalah untuk masalah-masalah khusus yang disebut juga dengan *toolbox*.

Di dalam Matlab terdapat beberapa *window* yang merupakan lingkungan kerja yang terpadu, dimana setiap *window* mempunyai kegunaan masing-masing. Ada beberapa *window* penting dalam Matlab yaitu :

- *Window* utama Matlab, *window* ini adalah *window* induk yang melingkupi seluruh lingkungan kerja Matlab, pada versi-versi rendah *window* ini secara khusus belum ada namun terintegrasi dengan *command window*.
- *Launc pad Window*, *window* ini mulai diperkenalkan pada versi 6 berfungsi bagi pemakai dalam memilih opsi dari fungsi dan *toolbox* yang ditawarkan Matlab.
- *Workspace window* yang berfungsi sebagai navigator bagi pemakai dalam penyediaan informasi mengenai variabel yang sedang aktif dalam *workspace* pada saat pemakaian, diperkenalkan pada versi 6.
- *Current directory window* merupakan fasilitas yang diperkenalkan pada versi 6 yang berfungsi sebagai *browser direktori* aktif yang hampir sama dengan *window explorer*.
- *Command window history* yang juga baru diperkenalkan pada versi 6 berfungsi sebagai penyimpan perintah-perintah yang pernah dikerjakan pada *workspace*.
- *Command window*, berfungsi sebagai penerima perintah dari pemakai untuk menjalankan fungsi-fungsi yang disediakan oleh Matlab. *Window* ini

merupakan inti dari Matlab yang menjadi media satu-satunya bagi pemakai untuk berinteraksi dengan Matlab.

- Matlab *editor window* yang berfungsi untuk membuat skip program Matlab, *window* ini mempunyai kemampuan untuk mendeteksi kesalahan pengetikan sintak oleh *programmer*.

Perintah-perintah matlab yang digunakan dalam menjalankan program yaitu :

1. *poly2trellis* :

   Fungsi *poly2trellis* menerima suatu *polynomial* dari *convolutional coding* dan dihubungkan kembali ke struktuk *trellis*.

   Contoh :

   trellis = poly2trellis([6 5],[33 45 0; 0 6 23])

   trellis =

   numInputSymbols : 4

   numOutputSymbols : 8

   numStates : 512

   nextStates : [512x4 *double*]

   outputs : [512x4 *double*]

   trellis.nextStates(1:6,:)

   ans =

   0 256 16 272

   0 256 16 272

1 257 17 273

1 257 17 273

2 258 18 274

2 258 18 274 **(2.2)**

2. RANDSRC

   Fungsi ini Menghasilkan matriks acak menggunakan abjad yang telah di tentukan dengan ukuran matriks M*N. Matriks ini merupakan matriks bipolar acak dimana -1 dan 1 mempunyai peluang yang sama untuk muncul.

   Contoh:

   Out = randsrc (2,3)

   Out =

   1 -1 -1

   -1 -1 1 **(2.3)**

3. PSKMOD (*phase shift keying* modulasi)

   Fungsi ini menentukan phase awal yang diinginkan dalam menjalankan simulasi. Nilai default dari INI_PHASE adalah 0. jika meinginkan phase yang berbeda tinggal diinputkan saja pada kolom ini_phase.

4. RECTPULSE (Rectangular pembentuk pulsa)

   Y = RECTPULSE (X, Nsamp) kembali Y, versi pulsa berbentuk persegi panjang X, dengan sampel Nsamp per simbol. Fungsi ini ulangan setiap simbol dalam X NSAMP kali.

5. Fungsi AWGN

   Y = AWGN (X, SNR, SIGPOWER) ketika SIGPOWER (*signal power*) numerik yang merupakan daya sinyal dalam dBW. Ketika SIGPOWER telah 'terukur', AWGN di ukur sebelum penambahan noise.

6. INTDUMP (*Integrated and dump*)

   Y = INTDUMP (X, NSAMP) mengintegrasikan sinyal X untuk 1 periode simbol, maka output satu nilai ke Y. NSAMP adalah jumlah sampel per simbol. Untuk sinyal dua dimensi, fungsi memperlakukan setiap kolom sebagai 1 saluran.

7. *PSK DEMOD*

   Z = PSKDEMOD (Y, M, INI_PHASE) menentukan tahap awal yang digunakan.

8. VITDECO

   Channel coding data biner menggunakan algoritma Viterbi.

9. FIND (Mencari indeks elemen nol)

   I = FIND(X) mengembalikan indeks linier dari array X yang nol.X mungkin merupakan ekspresi logika. untuk menghitung beberapa *subscript* dari indeks linear I, digunakan IND2SUB (I, UKURAN (X)).

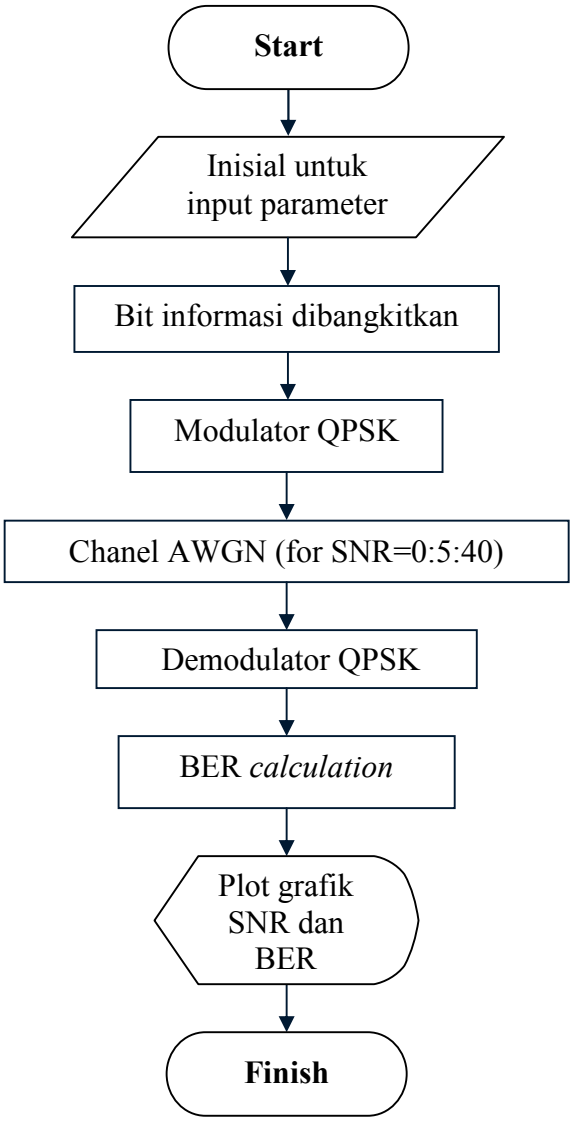
BAB III

PERANCANGAN DAN PEMODELAN

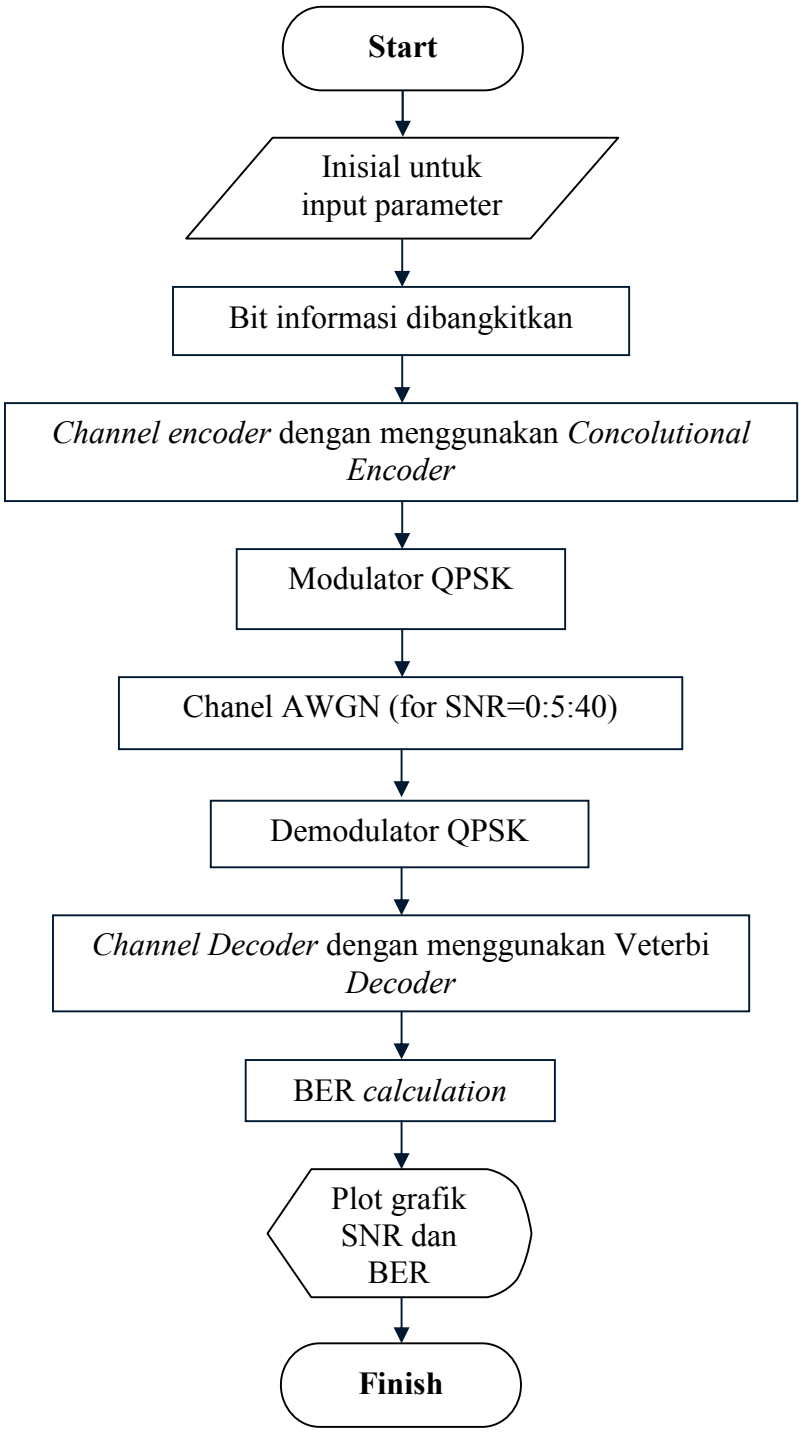
Pada tugas akhir ini program simulasi yang dibuat menggunakan bahasa pemograman komputer yang sangat diperlukan untuk menghitung dan menghasilkan data-data performansi yang akan dianalisa. Simulasi dilakukan karena sulitnya untuk melakukan perhitungan secara manual dan mahalnya peralatan pengukuran sinyal. Oleh karena itu, sistem yang akan diteliti direpresentasikan dalam sebuah model yang mendekati nilai sebenarnya. Maka model tersebut yang akan disimulasikan. Pada penelitian ini akan digunakan bahasa pemograman Matlab 7.1, sebab bahasa pemograman ini memiliki kemampuan yang mendukung untuk memproses data yang sangat banyak sebagai mana yang dibutuhkan pada simulasi ini.

3.1 *Flowchart* Simulasi

Langkah-langkah yang akan dilakukan yaitu berdasarkan *flowchart* simulasi pada Gambar 3.1 dan 3.2 di bawah ini :



Gambar 3.1 *Flowchart* Simulasi Tanpa menggunakan Teknik *Channel Coding* dan *Decoding*



Gambar 3.2 *Flowchart* Simulasi menggunakan Teknik *Channel Coding* dan *Decoding*

3.2 Perancangan Simulasi

Dalam perancangan simulasi ini ada banyak fungsi yang akan digunakan untuk memudahkan pemanggilan fungsi, masing-masing diberi nama sesuai dengan kegunaannya. Dalam perancangan simulasi ini menggunakan dua metode transmisi yaitu transmisi dengan menggunakan *Convolutional Code* dan *Algoritma Viterbi* dan transmisi tanpa menggunakan *Concolutional Code* dan *Algoritma Viterbi*.

3.2.1 Perancangan simulasi dengan penggunaan *Convolutional Code* dan *Algorima viterbi* pada transmisi

A. Data masukan acak berupa biner

- Bilangan acak yang terdistribusi yaitu antara 0 dan 1, jika bilangan acak yang dibangkitkan itu lebih kecil atau sama dengan 0,5 (< = 0,5) menghasilkan bit 0 dan jika bilangan acak yang dibangkitkan itu lebih besar dari 0,5 (> 0,5) menghasilkan bit 1. Bilangan acak ini masih dalam bentuk bilangan berkoma, jadi harus diubah dalam bentuk bilangan bulat. Bilangan bulat acak inilah yang akan menjadi *input* untuk *convolutional coding*.
- Menghasilkan data acak biner dengan jumlah data sebanyak NumSymb (3000 kolom) dan satu baris dan hasilnya disimpan pada *variable* dataIn yang kemudian sebagai *input* pada simulasi ini.

  dataIn = randsrc(NumSymb,1,0:1);

B. *Channel encoder* dengan menggunakan *Concolutional Encoder*

- *Channel coding* berfungsi untuk menjaga informasi atau data digital dari *error* yang mungkin terjadi selama proses transmisi dengan cara menambahkan bit redundansi (tambahan) ke dalam data yang akan dikirimkan. *Channel coding* beroperasi pada data digital dengan mengkodekan sumber informasi ke dalam urutan kode untuk ditransmisikan melalui kanal.
- Menghasilkan Trellis dari *Convolutional Encoder* dengan parameter panjang konstraint 4 dan 3 (*constlen*) dan generator code (*codegen*) dan hasilnya disimpan pada variable trellis.

  trellis = poly2trellis(constlen,codegen);
- Melakukan *Convolutional Encoding* dengan dataIn sebagai data input dan hasilnya disimpan pada *variable encoded*.

  [encoded]=convenco(dataIn,trellis);

C. Modulator QPSK

- Modulator merupakan bagian yang mengubah sinyal informasi kedalam sinyal pembawa (*Carrier*) dan siap untuk dikirimkan. Dalam modulator ini modulasi yang dibutuhkan adalah modulasi QPSK. Untuk mendapatkan modulasi QPSK mengatur sudut phasanya, karena QPSK mempunyai sudut phasa 0^0, 90^0, 180^0, dan 270^0.
- Memodulasi menggunakan modulator *Phase Shift Keying* (PSK) dengan orde 4 atau *Modulator Quadrature Phase Shift Keying* dan hasilnya disimpan pada variable qpsk_mod.

qpsk_mod = pskmod(encoded,M, pi/4);

- Melakukan filter (*Pulse Shaping Filter*) sebelum sinyal termodulasi dipancarkan ke kanal awgn, dan hasilnya disimpan pada variable qpsk_tx.

  qpsk_tx = rectpulse(qpsk_mod, Nsamp);

D. Chanel AWGN (for SNR=0:5:40)

Menambahkan AWGN pada sinyal termodulasi dengan kuat sinyal SNR sebagai parameter, dan hasilnya disimpan pada variable qpsk_*noise*.

qpsk_noise = awgn(qpsk_tx,SNR,'measured');

E. QPSK demodulator

- Demodulator adalah bagian yang memisahkan sinyal informasi (yang berisi data atau pesan) dari sinyal pembawa (*carrier*) yang diterima sehingga informasi tersebut dapat diterima dengan baik. Jika kita menginginkan kembali sinyal informasi yang di terima oleh penerima, maka kita harus mendeteksi kembali *phasa* tersebut yang telah melewati *channel* ber-*noise* (AWGN).
- Melakukan *Down Sampling* pada penerima untuk membalikan efek *filter* dan hasilnya disimpan pada variable qpsk_rx.

  qpsk_rx = intdump(qpsk_noise,Nsamp);

- Demodulasi menggunakan demodulator *Phase Shift Keying* dengan orde 4 atau Demodulator *Quadrature Phase Shift Keying* dan hasilnya disimpan pada variable qpsk_demod.

  qpsk_demod = pskdemod(qpsk_rx,M,pi/4);

F. Veterbi *Decoder*

- Algoritma Viterbi ini berfungsi sebagai pengkoreksi kesalahan yang terjadi pada data setelah dilakukan penyandian dengan *convolutional coding*. Prinsip dasarnya yaitu memilih lintasan yang mempunyai nilai kesalahan paling minimum. Pada Algoritma Viterbi ini menggunakan *hard-decision decoding*.
- Melakukan *Decoding* data yang diterima menggunakan veterbi *hard-decision decoding*.

  decoded= vitdeco(qpsk_demod,trellis,tblen,'term','hard');

G. BER *calculation*

- Mencari *errors*

  bit_errors = find(decoded ~= dataIn);

- Menghitung jumlah bit yang *error*

  NumErr = size(bit_errors,1);

- Menghitung laju bit *error* (BER) dengan cara jumlah bit *error* / jumlah bit yang dikirim

  BER(1,(SNR/5)+1) = NumErr/size(dataIn,1);

H. Plot grafik SNR dan BER

- Membuat jendela untuk grafik BER vs SNR

  figure('Name','Performansi Convolutional coding & Veterbi Decoding','NumberTitle','off','Toolbar','none','MenuBar',……

  'none');

- Memplot performansi sistem transmisi menggunakan *convolutional coding* dan veterbi *decoding* yang membandingkan BER dan SNR dalam sebuah kurva.

  semilogy (Snr,BER,'r-*');

3.2.2 Perancangan simulasi tanpa penggunaan *Convolutional Code* dan *Algorima viterbi* pada transmisi

A. Data masukan acak berupa biner

   Menghasilkan data acak biner dengan jumlah data sebanyak NumSymb (3000 kolom) dan satu baris dan hasilnya disimpan pada *variable* dataIn yang kemudian sebagai *input* pada simulasi ini.

   dataIn = randsrc(NumSymb,1,0:1);

C. Modulator QPSK

   Memodulasi menggunakan modulator *Phase Shift Keying* dengan orde 4 atau Modulator *Quadrature Phase Shift Keying* dan hasilnya disimpan pada variable qpsk_mod.

   qpsk_mod = pskmod(dataIn,M, pi/4);

D. Channel AWGN (for SNR=0:5:40)

   Menambahkan *adaptive white gaussian noise* pada sinyal termodulasi dengan kuat sinyal SNR sebagai parameter, dan hasilnya disimpan pada variable qpsk_*noise*.

   qpsk_noise = awgn(qpsk_mod,SNR,'measured',1234,'dB');

E. QPSK Demodulasi

Demodulasi menggunakan demodulator *Phase Shift Keying* dengan orde 4 atau Demodulator *Quadrature Phase Shift Keying* dan hasilnya disimpan pada variable qpsk_demod.

qpsk_demod = pskdemod(qpsk_noise,M,pi/4);

F. BER *calculation*

- Mencari *error*

  bit_errors = find(qpsk_demod ~= dataIn);

- Menghitung jumlah bit yang *error*

  NumErr = size(bit_errors,1);

- Menghitung laju bit *error* (BER) dengan cara jumlah bit *error*/jumlah bit yang dikirim.

  BER(1,(SNR/5)+1) = NumErr/size(dataIn,1);

G. Plot graphic for SNR verses BER

Memplot performansi *system* transmisi tanpa menggunakan *convolutional coding* dan veterbi *decoding* yang membandingkan BER dan SNR dalam sebuah kurva.

semilogy(Snr,BER,'b-o');

3.3 Parameter Pemograman

Parameter-parameter yang digunakan untuk menjalankan program :

1. Bit informasi

- Pembangkit bilangan acak

- NumSymb = 3000;
- M = 4; (Modulasi QPSK)

2. *Channel coding*
   - Tipe yang digunakan yaitu *convolutional code*
   - *k* (*input*) = log2(m)
   - *m* (*memory*) = 4
   - *constlen* (*constraint length*) = [4 3]
   - Code Rate = ½
   - *Code generator* = [4 5 17;7 4 2].
3. Modulasi dan demodulasi
   - Menggunakan modulasi dan demodulasi QPSK.
4. *Channel* ber-*noise*
   - Tipe yang digunakan *channel* AWGN
   - SNR yang digunakan 1 dB sampai dengan 40 dB
5. *Channel decoding*
   - tipe yang digunakan Algoritma Viterbi (*maximum likelihood*)
6. *Output*
   - *Output* yang diinginkan berupa grafik BER dan SNR.

BAB IV

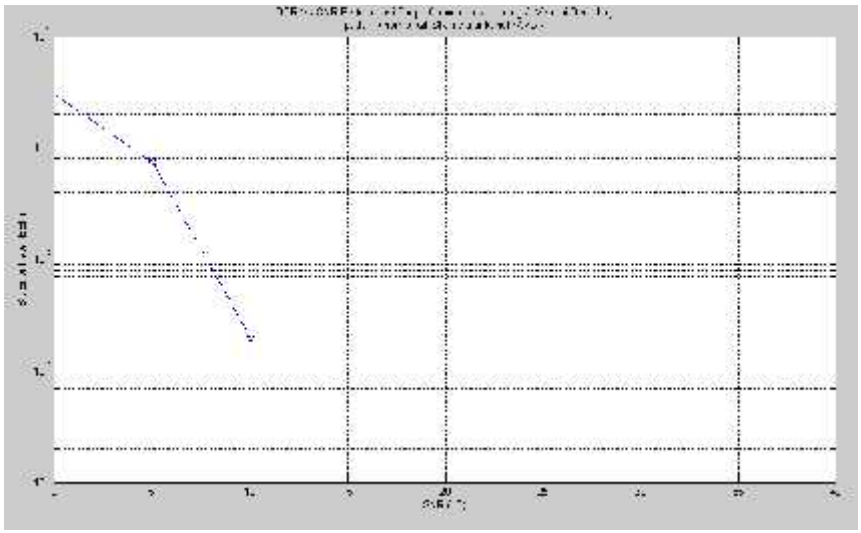
HASIL DAN ANALISA

4.1 Analisa Hasil Simulasi

Dari simulasi program pada bab 3 didapatkan beberapa grafik yang dapat menunjukkan pengaruh penggunaan *Convolutional Code* dan Algoritma Viterbi pada sistem telekomunikasi.

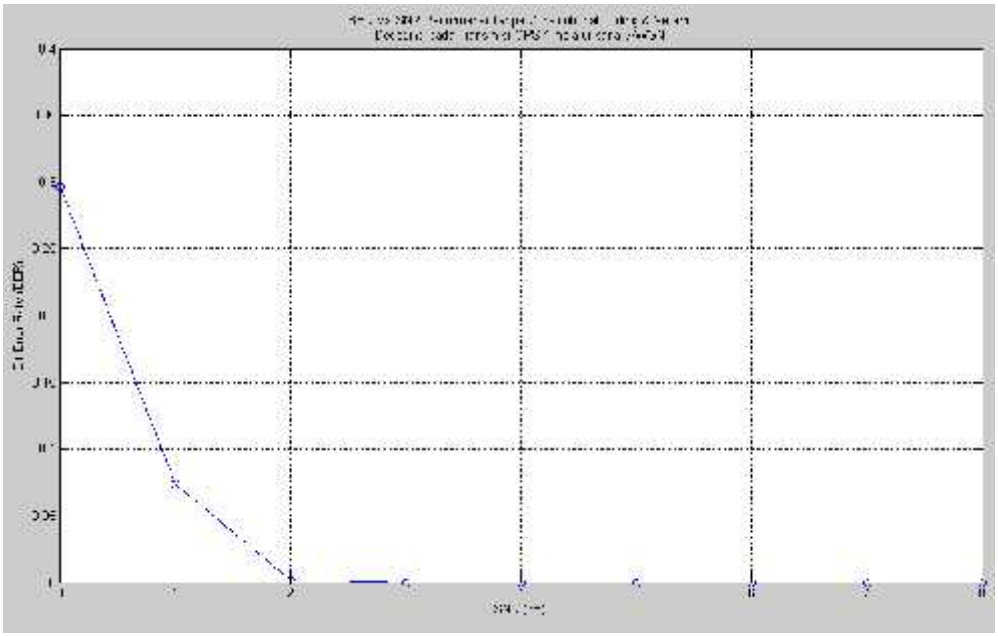
4.1.1 Hasil Simulasi Pada Transmisi Tanpa Penggunaan *Convolutional Code* dan Algoritma Viterbi

a. Grafik BER vs SNR dengan menggunakan Skala Linier untuk SNR dan Skala Logaritmik untuk BER (SNR = 0 : 5 : 40)



Gambar 4.1 BER Vs SNR pada transmisi tanpa *covolutional code* dan algoritma viterbi dengan menggunakan skala linier untuk SNR dan Skala Logaritmik untuk BER

b. Grafik BER vs SNR dengan menggunakan Skala Linier (SNR = 0 : 1 : 8)



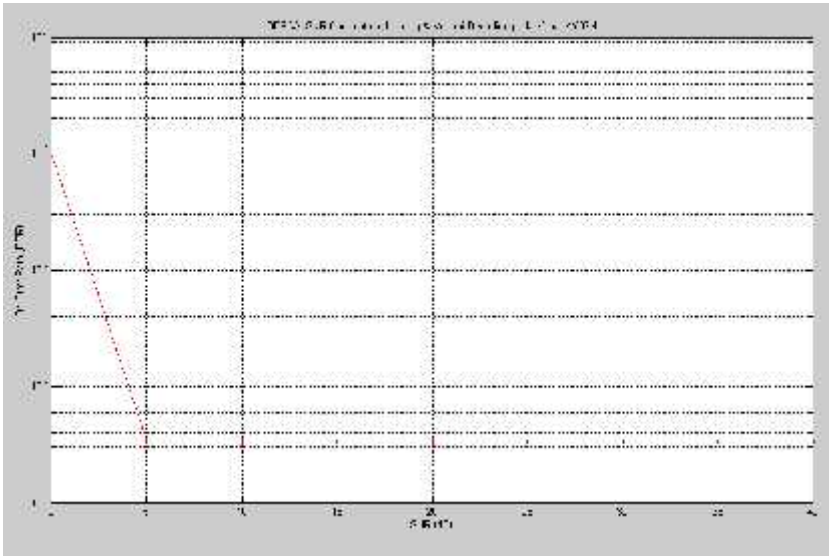
Gambar 4.2 BER Vs SNR pada transmisi tanpa *covolutional code* dan algoritma viterbi dengan skala linier

Dari gambar di atas pada grafik dapat dilihat tanpa *convolutional code* dan algoritma *viterbi* saat SNR 0dB BER-nya sangat besar yaitu mendekati 0.5 yang artinya separuh dari sinyal yang dikirim akan rusak atau cacat saat sampai di penerima. Sehingga sinyal yang dikirim tidak sesuai lagi dengan yang diterima dengan kata lain komunikasi tidak bisa dilakukan dengan kondisi tesebut. Maka cara yang bisa dilakukan adalah menambahkan daya atau memperbesar SNR menjadi 5dB, tapi dapat dilihat pada grafik penambahan SNR sebanyak 5dB tersebut, itu juga belum mendapatkan kualitas yang diinginkan karena BER masih sepersepuluh, yang artinya satu dari sepuluh bit yang dikirimkan oleh *transmitter,* hanya 9 bit yang akan diterima *receiver* tanpa cacat. Dalam hal ini BER masih dikategorikan tinggi dan akan merusak kualitas komunikasi. Dengan demikian

SNR harus dinaikkan lagi, sehingga saat SNR 10dB akan didapatkan nilai BER mendekati nilai 10^{-3}. Sebagaimana telah ditetapkan bahwa komunikasi dikatakan berkualitas baik jika memenuhi standar toleransi BER yaitu lebih kecil atau sama dengan 10^{-3} yang artinya dari seribu bit data yang dikirimkan hanya 1 bit yang diperbolehkan *error*. Dari hasil performansi dengan skala linier untuk nilai SNR dan skala logaritmik untuk BER, ketika SNR 0dB didapat BER-nya antara 10^{0} dan 10^{-1}, setelah dinaikkan SNR-nya 5dB maka nilai BER tersebut sudah mulai mengecil antara 10^{-1} dan 10^{-2}, dengan SNR 10dB nilai BER hampir mendekati nilai toleransi minimum, akan tetapi jika dinaikkan nilai SNR-nya lebih dari 10dB ternyata nilai BER-nya tidak mencapai nilai konstan. Jika performansi diganti dengan skala linier untuk kedua parameter SNR dan BER, untuk SNR di atas 10dB ternyata BER-nya berpengaruh dan mencapai nilai konstan. Dari hasil performansi tersebut *error* yang dihasilkan masih belum memadai. Tentu saja bila SNR diperbesar maka dipastikan daya dan biaya yang dibutuhkan akan lebih tinggi, karena untuk memperkuat sinyal akan diperlukan peralatan yang mahal dan daya yang lebih.

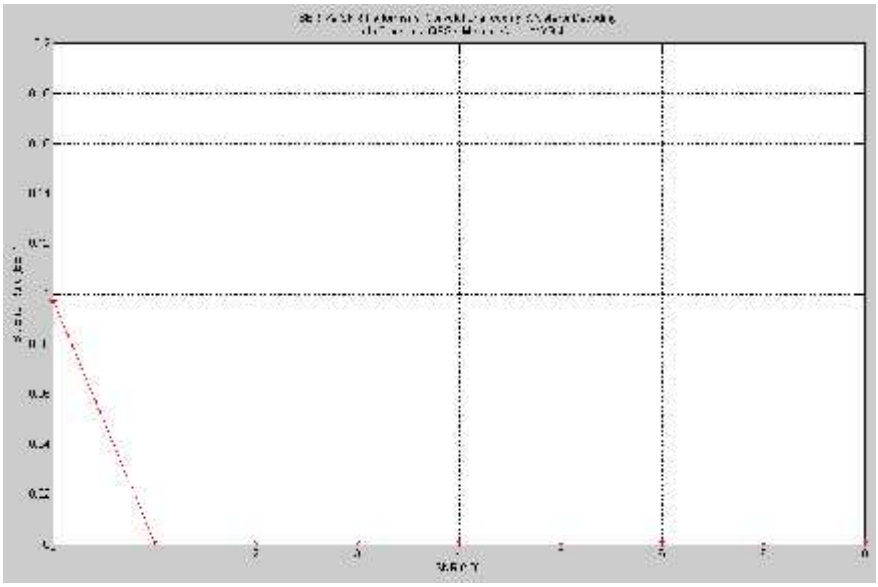
4.1.2 Hasil Simulasi Pada Transmisi Dengan Penggunaan *Convolutional Code* dan Algoritma Viterbi

a. Grafik BER vs SNR dengan menggunakan Skala Linier untuk SNR dan Skala Logaritmik untuk BER (SNR = 0 : 5 : 40)



Gambar 4.3 BER Vs SNR pada transmisi dengan *covolutional code* dan algoritma viterbi dengan menggunakan skala linier untuk SNR dan Skala Logaritmik untuk BER

b. Grafik BER vs SNR dengan menggunakan Skala Linier (SNR = 0 : 1 : 8)

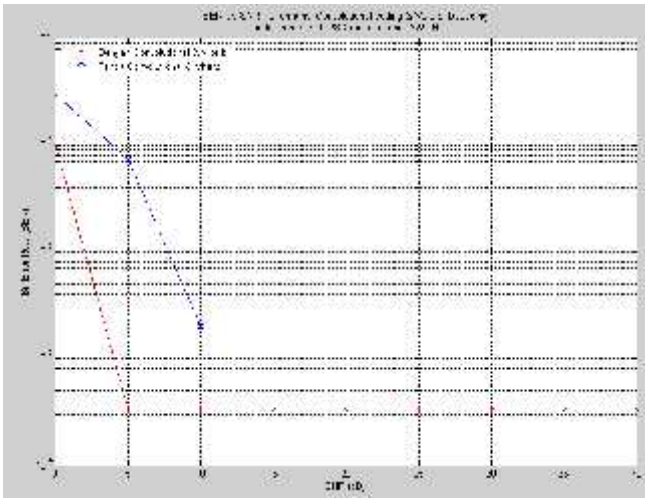


Gambar 4.4 BER Vs SNR pada transmisi dengan *covolutional code* dan algoritma dengan skala linier

Grafik di atas merupakan grafik sinyal yang menggunakan *convolutional code.* Dapat dilihat bahwa hanya dengan SNR 0dB nilai BER-nya hampir sama dengan nilai BER dengan SNR 5dB pada komunikasi tanpa *convolutional code*. Dengan menaikkan SNR menjadi 5dB maka komunikasi sudah mencapai nilai batas toleransi BER minimum yang telah ditetapkan. Sehingga akan tampak jelas perbandingan antara grafik tanpa menggunakan *covolutional code* dan algoritma viterbi dengan grafik yang menggunakan *convolutional code dan* algoritma viterbi.

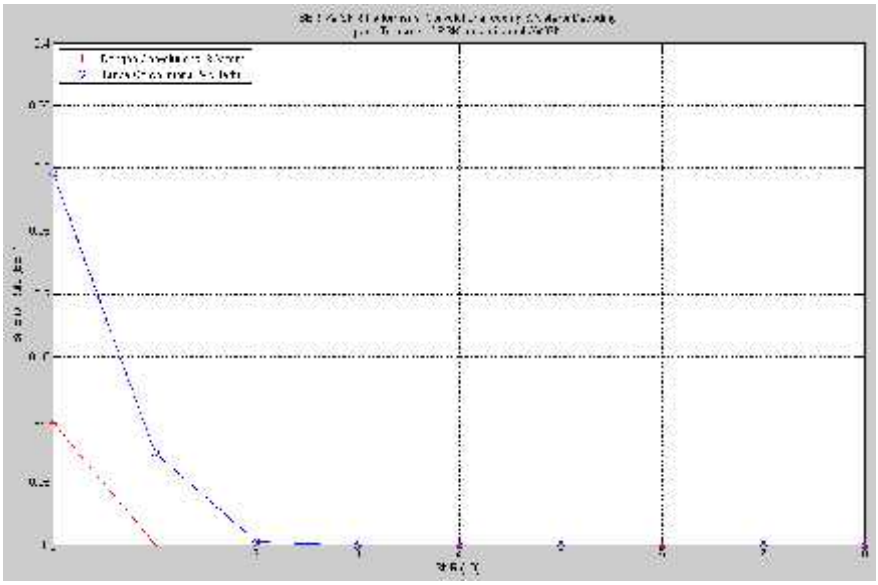
4.1.3 Perbandingan Hasil Simulasi Pada Transmisi Tanpa dan Dengan Penggunaan *Convolutional Code* dan Algoritma Viterbi

a. Grafik BER vs SNR dengan menggunakan Skala Linier untuk SNR dan Skala Logaritmik untuk BER (SRN = 0 : 5 : 40)



Gambar 4.5 Perbandingan BER Vs SNR pada transmisi tanpa dan dengan *convolutional code* dan algoritma viterbi dengan menggunakan skala linier untuk SNR dan skala logaritmik untuk BER

b. Grafik BER vs SNR dengan menggunakan Skala Linier (SNR = 0 : 1 : 8)



Gambar 4.6 Perbandingan BER Vs SNR pada transmisi tanpa dan dengan *convolutional code* dan algoritma Viterbi dengan skala linier

Perbandingan grafik BER Vs SNR pada kedua tipe transimisi akan sangat terlihat jelas penghematan daya pada transmisi dengan penggunaan *covolutional code* dan algoritma viterbi pada saat SNR 5dB BER yang terjadi berada pada kisaran 10^{-3} dan 10^{-4} artinya dengan SNR sebesar itu sudah didapatkan komunikasi yang layak. Sedangkan tanpa *convolutional code* dengan SNR yang sama yaitu 5dB BER yang terjadi masih sangat besar yaitu berada pada kisaran atau mendekati 10^{-1} tentu saja dengan nilai BER yang sangat besar ini komunikasi tidak memenuhi persyaratan karena banyaknya bit *error* yang diterima. Pada transmisi dengan *covolutional code*, minimum SNR yang dibutuhkan adalah 5dB untuk mendapatkan perfomansi sistem komunikasi yang baik. Untuk SNR di atas 5dB, penambahan nilai SNR tidak berpengaruh besar terhadap performansi, dan

nilai BER yang dihasilkan adalah konstan. Jadi pada transmisi ini hanya diperlukan SNR 5dB untuk mendapatkan kualitas komunikasi yang baik, dan tentu saja hal ini akan menghemat daya dan biaya. Sedangkan tanpa *covolutional code* justru peningkatan daya ini akan sangat membantu meningkatkan kualitas komunikasi karena dengan begitu nilai BER akan semakin kecil seiring dengan peningkatan SNR. Peningkatan SNR pastinya membutuhkan daya dan tambahan biaya lagi.

4.2 Analisa Penerapan

Pada dasarnya *Channel coding* berfungsi untuk menjaga informasi atau data digital dari *error* yang mungkin terjadi selama proses transmisi dengan cara menambahkan bit redundansi (tambahan) ke dalam data yang akan dikirimkan. Sedangkan *Convolutional code* merupakan salah satu jenis kode yang bisa mendeteksi dan mengoreksi *error* secara otomatis.

Channel Decoding berfungsi untuk memperkirakan *input* dari *coding* (informasi yang dikirimkan) dengan menggunakan aturan atau metoda tertentu yang menghasilkan kemungkinan jumlah *error* paling minimum. Sedangkan Algoritma Viterbi ini berfungsi sebagai pengkoreksi dan pendeteksi kesalahan yang terjadi pada data setelah dilakukan penyandian dengan *convolutional coding*.

Maka pada simulasi ini dapat dilihat penurunan nilai BER, karena pada *channel coding error* terlebih dahulu telah dideteksi dan diperbaiki sehingga pada akhirnya nilai BER akan sangat kecil dan bisa ditoleransi dan bahkan komunikasi sudah dikatakan berkualitas baik.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari tugas akhir ini dapat kita simpulkan bahwa :

1. Pada komunikasi dengan *covolutional code* dan algoritma viterbi hanya dibutuhkan SNR sebesar 5dB untuk mencapai standar minimum BER.
2. Saat SNR 5dB pada channel dengan *convolutional code* dan algoritma viterbi BER yang diperoleh sudah memenuhi standar yaitu antara 10^{-3} dan 10^{-4}. Sedangkan tanpa menggunakan *convolutional code* dan algoritma viterbi, BER-nya masih besar yaitu antara 10^{-1} dan 10^{-2}. Jadi jelas terlihat bahwa penggunaan *convolutional code* dan algoritma viterbi dapat meningkatkan performansi sistem.

5.2 Saran

Penelitian ini dapat dilanjutkan dengan membandingkan parameter lainnya, seperti *Amplitude error* dan *number error* disamping itu dapat juga dikembangkan dengan :

1. Penggunaan *trellis* diagram pada *convolutional code.*
2. Menggunakan teknik modulasi lainnya.
3. Membandingkan performansi dengan berbagai teknik modulasi lainnya.

Daftar Pustaka

Abdia, A gunaidi, "*shortcut of Matlab Programming",* Informatika, Bandung, 2006.

Emiyati, Islam dan suhermanto, "*Penggunaan Convolutional Coding pada Telemetry Channel Coding",* TIS 123-131, Lembaga Penerbangan dan Antariksa Nasional, Jakarta, 2005.

Fleming, Chip, " *A Tutorial on Convolutional Coding with Viterbi Decoding* ", November. 2002. [Online] Available http://home.netcom.com/~chip.f/viterbi/tutorial.html, diakses tanggal 20 Maret 2009.

H. Robert, " *Convolutional Codes dan Algoritma Viterbi* " Agustus. 2008. [Online] Available http://the-art-of-ecc.com/5_Convolutional/index.html, diakses tanggal 20 Maret 2009.

Langton, Charan, " *Signal processing and simulation newsletter* " Juli. 1999. [Online] Available http://www.complextoreal.com/convo.html, diakses tanggal 20 Maret 2009.

Ganesha, Esha, "*Implementasi Pengawasandian Viterbi dengan Field Programmable Logic Array (FPGA)*, Universitas Gajah Mada (UGM), jogjakarta, 2007.

Liu, Er, " *Convolutional Coding & Viterbi Algorithm* ", Page 1-17, Helsinki University of Technology, Helsinki, 2004.

Nassar, Carl, " *Telecommunications Demystified* ", Chapter 7, Halaman 197-217, LLH Technology Publishing, Eagle Rock, Virginia, 2001.

Nugroho, " *Convolutional* " http://bsnugroho.googlepage.com/convolutional, diakses tanggal 02 Maret 2009.

Pillai, Krishna, " *Convolutional Code* ", Januari. 2009. [Online] Available http://www.dsplog.com/id/2009/01/04/convolutional-code/, diakses tanggal 02 Maret 2009.

_____, " *Viterbi Decoder* ", Januari. 2009. [Online] Available http://www.dsplog.com/id/2009/01/04/viterbi/, diakses tanggal 02 Maret 2009.

Purnamirza, Teddy, " *Modul Praktikum Matlab* ", Teknik Elektro UIN Suska, Pekanbaru, 2006.

_____, " *Analisa Performansi OFDM Pada Kanal Daerah Perkotaan, Perdesaan dan Terbuka* " UIN suska, Pekanbaru, 2007.

Toussaint, T. Godfried, Prof. " *Algorithm Viterbi in Confession Text by Luz Abril Torres Mendez* ", 2000. [Online] Available http://www.cim.mcgill.ca/~latorres/Viterbi/va_alg.html, diakses tanggal 20 Maret 2009.

Utomo, pramudi, dkk, " *Teknik telekomunikasi jilid 2* ", bagian 9, halaman 179-199, Direktorat Pembinaan Sekolah Menengah Kejuruan Departemen Pendidikan Nasional, 2008.